Technical Report ICMA-94-187

ON THE COMPUTATION OF SIMPLICAL APPROXIMATIONS OF IMPLICITLY DEFINED TWO-DIMENSIONAL MANIFOLDS

by

Monica L. Brodzik
and
Werner C. Rheinboldt

April, 1994

ICMA

Department of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15260

On the Computation of Simplicial Approximations of Implicitly Defined Two-dimensional Manifolds *

Monica L. Brodzik and Werner C. Rheinboldt

Department of Mathematics and Statistics
University of Pittsburgh, Pittsburgh, PA 15260

Abstract: *A method is presented for the computation of a simplicial approximation covering a specified subset M_0 of a two-dimensional manifold M in R^n defined implicitly as the solution set of a nonlinear system $F(x) = 0$ of $n-2$ equations in n unknowns. The given subset $M_0 \subset M$ is the intersection of M with some polyhedral domain in R^n and is assumed to be bounded and non-empty. The method represents an extension of a local simplicial approximation process developed earlier by the second author.*

1. Introduction

For nonlinear mappings $F : R^n \to R^m$, $n = m + d$, $d \geq 2$, natural conditions exist that guarantee the solution set

$$M = \{x \in R^n; F(x) = 0\} \tag{1.1}$$

to have the structure of a differentiable, d-dimensional manifold M in R^n. We present here an algorithm for computing a simplicial approximation that covers an a priori specified subset of such an implicitly defined, two-dimensional manifold.

In computer aided geometric design (CAGD) and related applications, manifolds are often parametrically defined; that is, as the image set $\{x \in R^n; x = \Phi(u, v)\}$ of some known parametrization mapping $\Phi : R^2 \to R^n$. In this case, various triangulation methods have been proposed, (see e.g. [C93] and the survey [BE92]) all of which represent, in essence, extensions of techniques developed for the triangulation of flat spaces.

So far the case of implicitly defined manifolds (1.1) has not received as much attention. The earliest papers appear to be [AS84], [AS85], and [AG87]; they use a piecewise linear, combinatorial continuation algorithm to construct a simplicial complex in the ambient space R^n that encloses the implicitly given d-dimensional manifold. The barycenters of appropriate faces of the enclosing simplices are then chosen to compute a global, piecewise linear approximation to the manifold. However, since, in general, the resulting vertices do not lie on the manifold, this does not represent a simplicial approximation in the standard sense of combinatorial topology.

A first method for the direct computation of local pieces of a simplicial approximation of the manifold M of (1.1) was presented in [R87], [R88]. There standardized patches of triangulations of the tangent spaces T_xM of M are projected onto the manifold by smoothly

* This work was supported in part by ONR-grant N-00014-90-J-1025, and NSF-grant CCR-9203488

varying projections constructed by a moving frame algorithm. The method is applicable to manifolds of any dimension $d \geq 2$ but was used principally in the case $d = 2$. In [Ho91] a modified version of this method for the case $d = 2$ was considered. It uses interpolating polynomials to predict new points; but, before correcting them onto the manifold, a search is conducted to determine whether the new points may cause a potential overlap with any earlier computed, existing triangle. If such an overlap is detected, the predicted points are identified with appropriate nearby existing vertices.

Two new methods were developed in [MM93] and [He93]. In both cases, interest does not center on the explicit construction of a simplicial approximation of the implicitly given, 2-dimensional manifold (1.1), but on tessellating it by a cell-complex. In [He93] the manifold is covered by overlapping ellipsoidal cells each of which is obtained as the projection of a suitable ellipse on some tangent space. In [MM93] a complex of non-overlapping cells with piecewise curved boundaries is constructed by tracing a fish-scale pattern of one-dimensional paths on the manifold. Both of these methods appear to be intrinsically designed for 2-dimensional manifolds.

Here we present an extension of the original method given in [R88]. In particular, the process is globalized to allow for the computation of a simplicial approximation that covers a specified domain of the manifold. The algorithm is developed for the case $d = 2$ but our aim was to use tools that, in principle, can be generalized to higher dimensional manifolds. For this purpose, the mentioned moving frame algorithm is replaced by a careful consideration of the orientation of the triangles. This eliminates the calculation of d-dimensional singular value decompositions which can become costly when working with manifolds of dimension higher than two.

In the original algorithm for the case $d = 2$, the patch that is projected from the tangent space T_xM at $x \in M$ onto M always consists of a hexagonal neighborhood of six triangles centered at x. This was feasible since the algorithm was only applied locally. But, such a fixed patch is likely to cause local overlaps when the algorithm is applied to larger domains of M. Thus in the new method the patches are constructed adaptively and are allowed to have fewer than six triangles.

The algorithm works with an advancing front technique. We begin with a point $x_0 \in M$ and add it to the database that stores the triangulation. Then, in analogy with the original method, a first hexagonal neighborhood around x_0 is constructed in the tangent space $T_{x_0}M$ and its vertices are projected onto M and added to the data base. The starting front of the process is formed by those of the six new vertices that are contained in the interior of the given domain of M. The others are marked as exterior vertices. In general, a step of the method consists in the selection of a point x_c on the current front. Then the existing triangles incident with x_c are projected onto the tangent space $T_{x_c}M$. This results in a partial neighborhood of x_c with a gap that still has to be closed. If the gap is too small, it is closed by identifying its two open edges, otherwise, it is divided into an optimal number of triangles. The resulting new points in $T_{x_c}M$ are then projected onto M and added to the data base. If these points belong to the given domain of M they are also added to the front, otherwise they are tagged as exterior points. Finally the step is completed by removing the current frontal point x_c from the front, since it now has a complete neighborhood of triangles. The process terminates when the front is empty.

The resulting simplicial approximation covers the given domain. It is not difficult to adjust all exterior points onto the boundary of the domain although this may result in needle-like triangles. More advantageous is to use a local Delauney-type method to effect the adjustment onto the boundary of the domain. More generally, such a Delauney approach can serve also to improve the quality of the entire mesh. We shall not enter into the details here. As noted earlier, our aim is to extend the method to implicitly defined manifolds of dimension larger than two. This is the topic of ongoing work.

In Section 2 below we give a brief summary of some background material needed for the development of the method. Then, Section 3 outlines the data structure used here, and Section 4 presents a detailed description of the algorithm. Finally, Section 5 shows several numerical examples.

2. Background

Throughout the paper $F : \mathcal{D} \subset R^n \mapsto R^m$, $n = m + d$, $d = 2$, is a given nonlinear mapping of class C^1 on the open, connected domain $\mathcal{D}$ for which the first derivative $DF(x)$ has rank m for all $x \in \mathcal{D}$. Then is is well-known that the solution set

$$M = \{x \in \mathcal{D};\ F(x) = 0\}, \tag{2.1}$$

is a two-dimensional C^1-manifold in R^n without boundary (see, e.g., [S79], [R86]). At any $x \in M$ the tangent space T_xM is identified with $\ker DF(x)$.

For the definition of the subset of M that is to be triangulated we introduce the hyperplanes

$$H_k = \{x \in R^n;\ b_k^T(x - p_k) = 0\}, \quad k = 1, \dots n_h, \tag{2.2}$$

where $b_k \in R^n$ is a unit normal vector of H_k and $p_k \in R^n$ a point in H_k. Then, with the corresponding half-spaces

$$S_k = \{x \in R^n;\ b_k^T(x - p_k) \geq 0\}, \quad k = 1, \dots n_h,$$

the set $S = \cap S_k$ is a polyhedral domain in R^n. The desired subset of M will be the intersection $M_0 = S \cap M$. We assume always that M_0 is a bounded set with a non-empty relative interior. Points in the relative interior of M_0 will be called interior points of M_0 while all others are designated as exterior points.

As in [R86] we introduce at any "current" point $x_c \in M$ a tangential local coordinate system. For this, let the columns of $U^c \in R^{n\times 2}$ define an orthonormal basis of the tangent space $T_{x_c}M$ at x_c. Then the implicit function theorem applied to the equation

$$F(x_c + U^c y + DF(x_c)^T z) = 0, \qquad y \in R^2, \quad z \in R^m, \tag{2.3}$$

guarantees the existence of open neighborhoods $\mathcal{U}_c$ of the origin of R^2 and $V_c \in R^n$ of x_c, respectively, such that for any $y \in \mathcal{U}_c$ there exists exactly one solution z of (2.3) with $x_c + U^c y + DF(x_c)^T z \in V_c$ and that the mapping $\psi : \mathcal{U}_c \mapsto R^m$, $\psi(y) = z$, is of class C^1 on $\mathcal{U}_c$. Evidently, we have $\psi(0) = 0$ and $D\psi(0) = 0$ and

$$\Phi : \mathcal{U}_c \mapsto R^n, \quad \Phi(y) = x_c + U^c y + DF(x_c)^T \psi(y), \quad \forall y \in \mathcal{U}_c, \tag{2.4}$$

is a diffeomorphism from U_c onto $M \cap V_c$. In other words, Φ^{-1} is a chart of M at x_c and we call Φ a tangential local coordinate map at x_c.

The evaluation of $\Phi(y)$ is equivalent to projecting the part $x = (x_c + U^c y) \in T_{x_c}M$ orthogonal to $T_{x_c}M$ onto M; that is, to solving the system

$$\begin{aligned} F(x) &= 0 \\ U^{cT}(x - (x_c + U^c y)) &= 0. \end{aligned} \tag{2.5}$$

Thus in general we require, for any point $x \in R^n$ sufficiently close to x_c, the capability of projecting x onto M orthogonally to $T_{x_c}M$. There are several ways of doing this. For example, as in [R88], we may use the QR-factorization

$$DF(x_c)^T = (Q_1, Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix} \tag{2.6}$$

where rge Q_2 = ker$DF(x_c)$, and set $U^c = Q_2$. Then the system in (2.5) can be solved by using the chord Gauss-Newton method

$$x_{k+1} = x_k - DF(x_c)^T(DF(x_c)DF(x_c)^T)^{-1}F(x_k), \quad k = 1, 2, \ldots$$

The following algorithm incorporates two possibilities for providing x, namely, (i) $x \in R^n$, $x \neq x_c$ is directly given, or (ii) $y \in R^2$ is provided and $x = x_c + U^c y$ is still to be computed. The two choices are passed to the algorithm via the input variable job, with $job = 1$ for case (i), and $job = 2$ for (ii).

```
Proj:  Input: $x_c$, $x$, $y$, $R$, $Q_1$, $Q_2$, $job$;
    if '$job = 1$'  then  $x := x$  else  $x := x_c + Q_2 y$  end
    while 'no convergence'
        solve  $R^T z = F(x)$  for  $z$
        $x := x - Q_1 z$
    Output: $x$.
```

For all x sufficiently near x_c the process converges to a unique $x^* \in M$, see e.g. [DH79].

3. Data Structure and Problem Definition

Any mesh generation algorithm depends critically on the data structure, for which there are many choices. Since we restrict ourselves here to two-dimensional manifolds, where the relationships between nodes and simplices is fairly straightforward, we chose a simple data structure consisting of three two-dimensional arrays **xnod**, **sim**, and **nod**, for the node-coordinates, the simplex/node incidences, and the node/simplex incidences, respectively. Their organization is summarized in Table 3.1. In the i-th row of the **sim** array, the indices of the three vertex nodes of the i-th simplex are stored in an order that defines the simplex's orientation. Any two consecutive simplices listed in the i-th row of the **nod** array share an edge.

Array	*Type*	*Dimension*	*Contents of row i*
xnod	Double Precision	*maxnod* rows *nvar* columns	**xnod**$(i,j) = j^{th}$ coordinate of node i, $j = 1,\dots,nvar$
sim	Integer	*maxsim* rows 3 Columns	**sim**(i,j) = index of j^{th} vertex of simplex i, $j = 1,\dots,3$
nod	Integer	*maxnod* rows 7 columns	**nod**(i,j) =index of j^{th} simplex incident with node i, $j = 1,\dots,k \leq 6$ **nod**$(i,j) = 0$, $j = k+1,\dots,6$ **nod**$(i,7)$ = *nodtyp* (See later)

Table 3.1: Data Structure

The operations defined on the database are as follows:

Addnod[x, *nodtyp*]:
Stores coordinates of a new point x in the next available row of **xnod**, and enters the point's *nodtyp* in the next available row of **nod**.

Addsim[x_1, x_2, x_3]:
Adds a new simplex to the **sim** array, and updates the **nod** array by adding the simplex's index to the rows corresponding to the vertices x_1, x_2, x_3.

Equate[x, x_1, x_2, $\bar{x}$]:
Identifies two computed points x_1, x_2, which are incident at x, by replacing x_1 with the projected average $\bar{x}$ of x_1 and x_2, removing x_2, and then updating all three tables of the data structure.

Neighb[x]:
Checks if the given point x is connected to a point which is exterior to M_0.

Note that in this data structure all the details about the actual data storage and manipulation had to be included in the software package. This is here not a great disadvantage since for two-dimensional manifolds these details are fairly simple, and our resulting data manipulation software has shown to be acceptably fast. However, when generalizing our triangulation algorithm to higher-dimensional manifolds, we will use the relational database management system SQL to keep track of all the details of the data storage, since the relationships between nodes and simplices are then much more complicated.

The user is assumed to supply the following three subroutines defining the problem:

Fct[x, $F(x)$]:
Defines the function F in (2.1) and returns the components of $F(x)$ evaluated at the given point x.

Dfct[x, $DF(x)$]:
Defines the Jacobian DF of F and returns the components of $DF(x)$ evaluated at the given point x.

Bnds[x, k, b_k, p_k]:
Defines the hyperplanes in (2.2) and returns the components of the vectors b_k and p_k for a given $k \in \{1, \dots, n_h\}$.

For a given point $x \in R^n$, the following algorithm **Chkbnd** determines whether or not x belongs to the polyhedral domain S. In addition, it computes the distance $dmin$ between x and the nearest hyperplane H_{knear}.

```
Chkbnd: Input: $x$, number of hyperplanes $n_k$;
    for $k = 1, n_k$
        $\{p_k, d_k\}$ = Bnds[$k$]   /*Get $p_k \in H_k$, unit vector $b_k$ normal to $H_k$.*/
        $d = b_k^T(x - p_k)$   /*Compute signed Euclidean distance $d$ from $x$ to $H_k$.*/
        if $d \leq 0$ then   /*$x$ does belong to $S$:*/
            $knear := k$,  $dmin := d$
            return
        else
            if $k = 1$ then
                $knear := 1$,  $dmin := d$   /*Initialize.*/
            else if $d < dmin$ then
                $knear := k$,  $dmin := d$
            end
        end
    end
    Output: $knear$, $dmin$.
```

4. The Triangulation Algorithm

The triangulation of the subset $M_0 = S \cap M$ of the manifold M begins with a user-supplied starting point $x_0 \in R^n$ which need not be on M. The process calculates the QR-factorization (2.6) of $DF(x_0)$ and uses the routine **Proj**, with $x := x_0$ and $job := 1$, to project x_0 onto a point $x_m \in M_0$. If **Proj** fails, the user is requested to supply a different starting point. Otherwise, each of the six vertices

$$(\frac{h\sqrt{3}}{2}, \frac{h}{2}), \quad (0, h), \quad (\frac{-h\sqrt{3}}{2}, \frac{h}{2}), \quad (\frac{-h\sqrt{3}}{2}, \frac{-h}{2}), \quad (0, -h), \quad (\frac{h\sqrt{3}}{2}, \frac{-h}{2})$$

of the hexagonal neighborhood of equilateral triangles around the origin in R^2 is mapped onto the affine tangent space $x_m + T_{x_m}M$ and then projected onto M, using again **Proj**. For h either a user-supplied step size or a smaller one is used whichever guarantees successful projection of the first of the six points onto M. (This first successful value of h is retained as the constant step size throughout the remainder of the triangulation). The projected points inherit the connectivity pattern of the original hexagonal neighborhood of equilateral triangles in R^2.

Generally, as mentioned earlier, a point of **nod** is either an interior point (of M_0) or an exterior point. This can be determined by means of **Chkbnd**. Any interior point is identified as a frontal point if it does not yet have a completed simplicial neighborhood.

Accordingly, each point of **nod** is classified by assigning it a node type as shown in Table 4.1. The node type is updated as the triangulation progresses.

nodtyp	*Definitio*
-1	Frontal point still to be handled.
0	Interior point connected only to interior points.
1	Interior point connected to an exterior point.
2	Exterior point connected to an interior point.

Table 4.1: Types of nodes in M.

The frontal points, in their order in **nod**, form a queue. At each step of the process the topmost point is removed from this queue and, when new points are computed new frontal points may be added at the end. The process stops when the queue is empty.

Let x_c denote the next frontal point taken from the queue. Note that, after the correction of the starting point, x_c is not equal to x_0. By definition a portion of the simplicial neighborhood of x_c has already been calculated and stored. Figure 4.1 shows a typical example of this existing part, and Figure 4.2 gives a view of its local representation in R^2. The main tasks are now to determine the gap in the simplicial neighborhood that still needs to be closed, to decide how to close it, and finally to close it.

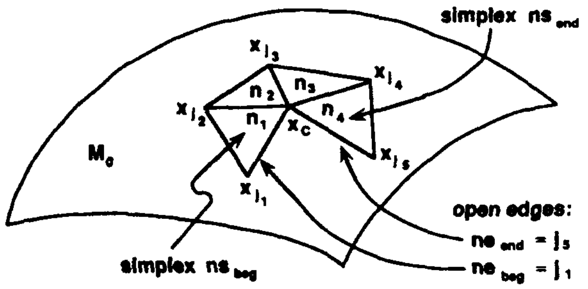

FIGURE 4.1

From the data structure the two simplices ns_{beg} and ns_{end} are immediately known. It is also fairly easy to find the edge-defining nodes $x_{j_1}, \ldots, x_{j_{k+1}}$ in the same order of consecutive appearance around x_c as the incident simplices appear in **nod**. Then the open edges are simply $ne_{beg} := j_1$ and $ne_{end} := j_{k+1}$. However, it is not immediately known whether the incident simplices $n_1, \ldots, n_k$ (with $n_m :=$ **nod**(i,m), $m = 1, \ldots, k$) appear clockwise, as in Figure 4.1, or counterclockwise around x_c when projected back onto $T_{x_c}M$. Knowledge of this ordering around x_c is essential in determining whether the gap is the clockwise angle from ne_{beg} to ne_{end} or its complement.

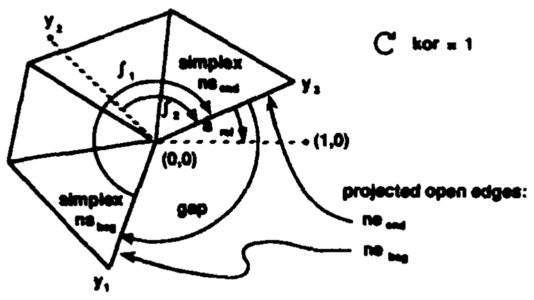

FIGURE 4.2

Central to the determination of this ordering is the unit average direction vector $\bar{x}$ defined by

$$a = \frac{1}{k+1}\sum_{i=1}^{k+1}(x_{j_i} - x_c), \quad \bar{x} := \frac{a}{\|a\|}$$

The gap is then obtained as follows. For $\bar{x}$ and the normalized open edge directions

$$x_1 := \frac{(x_{j_1} - x_c)}{\|(x_{j_1} - x_c)\|}, \qquad x_{k+1} := \frac{(x_{j_{k+1}} - x_c)}{\|(x_{j_{k+1}} - x_c)\|},$$

determine the local coordinates $y_1 := U^T x_1$, $y_2 := U^T \bar{x}$, and $y_3 := U^T x_{k+1}$ in R^2. It is expected that y_2 will point in a direction which is in the complement of the gap angle between y_1 and y_3. (See Figure 4.2). The reference clockwise rotation angle α_{ref} of y_3 into $e_1 := (1,0)^T$ is calculated by the following Givens-type algorithm:

```
Angle: Input: vector $(a,b)$;
    $nrm := \|(a,b)\|$   /* Euclidean norm of $(a,b)$.*/
    if $nrm = 0$  then $\alpha := 0$; return  endif
    if $abs(a) \geq abs(b)$ then
        if $b = 0$  then $\phi := 0$  else $\phi := arctan(\frac{b}{a})$  endif
        if $a \geq 0$ then
            $\alpha := \phi$
            if $b < 0$  then $\alpha := 2\pi - \phi$  endif
        else
            $\alpha := \pi - \phi$
            if $b < 0$  then $\alpha := \pi + \phi$  endif
        endif
    else
        if $a = 0$ then
            $\phi := 0$
    else
        $\phi := arctan(\frac{b}{a})$
        if $a < 0$   then $\phi := -\phi$   endif
    endif
    if $b \geq 0$   then $\alpha := \frac{\pi}{2} - \phi$   else $\alpha := \frac{3\pi}{2} + \phi$   endif
end
Output: $\alpha$, $nrm$.
```

Let A denote the 2x2 matrix which effects a clockwise rotation by α_{ref}. Then Ay_1 and Ay_2 represent clockwise rotations by α_{ref}, and the clockwise rotation angles β_1 and β_2 of the resulting vectors Ay_1 and Ay_2 into e_1, respectively, are, effectively, the clockwise rotation angles from y_1 and y_2 into y_3. (See Figure 4.2). If $\beta_2 < \beta_1$, then the simplices $n_1, \ldots, n_k$ are arranged clockwise from n_1 to n_k around the origin in the local coordinate system. Hence the gap angle's magnitude is $2\pi - \beta_1$, and its orientation indicator kor is defined to be 1. Otherwise if $\beta_2 > \beta_1$, then the simplices are arranged counterclockwise, the gap angle is β_1, and we set $kor = 2$. The following algorithm implements this gap determination process:

```
Agap:  Input: center point $x$, tangent basis $U = [u_1, u_2]$ at $x$;
    Order $x_{j_1}, \ldots, x_{j_p}$   /*Order nodes incident at $x$, so $x_{j_1}$, $x_{j_p}$ define open edges.*/
    $ne_{beg} := j_1$,  $ne_{end} := j_p$   /*Define indices of open edges.*/
    Find $ns_{beg}$, $ns_{end}$   /*Get indices of simplices containing open edges.*/
    $\bar{x} = \sum_{k=1}^{p}(x_{j_k} - x)$,   $\bar{x} := \bar{x}/\|\bar{x}\|$   /*Get unit average direction vector.*/
    $x_k := (x_{j_k} - x)/\|x_{j_k} - x\|$,   $k = 1, p$   /*Normalize open edge directions.*/
    $y_1 := U^T x_1$   /*Get the $R^2$ local coordinate vector of $x_1$.*/
    $y_2 := U^T \bar{x}$   /*Get the $R^2$ local coordinate vector of $\bar{x}$.*/
    $y_3 := U^T x_p$   /*Get the $R^2$ local coordinate vector of $x_p$.*/
    $\alpha :=$Angle$[y_3]$   /*Get clockwise rotation angle $\alpha$ of $y_3$ into $e_1 := (1,0)^T$.*/

    $A := \begin{pmatrix} cos\ \alpha & sin\ \alpha \\ -sin\ \alpha & cos\ \alpha \end{pmatrix}$

    $y_k := Ay_k$,  $k = 1, 2$   /*Rotate $y_1$ and $y_2$ clockwise by $\alpha$.*/
    $\beta_k :=$Angle$[y_k]$   /*Get clockwise rotation angle $\beta_k$ of $y_k$ into $e_1$, $k = 1, 2$.*/
    if $\beta_2 < \beta_1$ then
        $gap := 2\pi - \beta_1$   /*Define $gap$ angle which needs to be closed.*/
        $kor := 1$   /*Define orientation $kor$ of $gap$. */
    else
        $gap := \beta_1$
        $kor := 2$
    end
    Output: $gap$, $kor$, $\alpha$, $ne_{beg}$, $ne_{end}$, $ns_{beg}$, $ns_{end}$.
```

Once the gap angle gap and its corresponding kor value have been determined, the process of closing the gap depends on the magnitude of gap, the number of already existing simplices n_{old} incident at x_c, and a fixed minimum acceptable angle size α_{min}. If the gap is small; that is, if $gap < \alpha_{min}$, then the following merge algorithm closes the gap by "identifying" the open edges j_1 and j_{k+1} and then replacing the nodes defining these edges in **xnod** by a projection of their average $\frac{1}{2}(x_{j_1} + x_{j_{k+1}})$ onto M.

```
Merge: Input: center point $x$; other endpoints $x_1$, $x_2$ defining open edges;
    $\bar{x} := \frac{1}{2}(x_1 + x_2)$
    $\bar{x}_m$ =Proj[$\bar{x}, \bar{x}, \bar{x}, R, Q_1, job = 1$]   /*Project $\bar{x}$ onto $M$.*/
    Call Equate[$x, x_1, x_2, \bar{x}_m$]   /*Replace $x_1, x_2$ with $\bar{x}_m$ in data structure.*/
```

If $gap \geq \alpha_{min}$ and n_{old} already equals the maximum allowable value of 6, then the algorithm stops with an error return. Otherwise,

$$n_{new} := max \{ \, i \; : \; \frac{gap}{i} \geq \alpha_{min}, \; 1 \leq i \leq (6 - n_{old}) \, \}$$

new simplices are added to close the gap at x_c and hence to complete the neighborhood of simplices around x_c. If $n_{new} = 1$, then only one simplex is added, namely the one defined by the already existing points x_c, x_{j_1}, and $x_{j_{k+1}}$. The indices of these points are entered into the next row of the **sim** array in an order such that the orientation of the new simplex agrees with that of its adjacent simplex ns_{end}. If $n_{new} > 1$, then $n_{new} - 1$ new points are needed to define the new simplices that close the gap. In this case, the gap - angle is divided into n_{new} equal angles, and the new points in the local coordinate system are defined to be in the resulting directions and at a distance h from x_c. One at a time, in order of rotation from y_3 into the gap, these points in $x_c + T_{x_c}M$ are constructed, projected onto M by **Proj**, and added to the database by **Addnod**. As each new point on M is found, **Addsim** adds to the database the simplex defined by the new point and the endpoints of the open edge from which it was rotated. At the last new point $x \in M$ in the gap, a second simplex is added, namely the one formed by x, x_c, and x_{j_1}. This simplex completes the neighborhood around x_c.

For some new direction **t**, **Proj** may fail to project the tangent point $x_c + h\mathbf{t}$ onto M. In this case, a simple continuation process is started along the direction **t** in the following way. A temporarily smaller stepsize $h := \frac{h}{2^k}$, $k = 1, 2, \ldots$ is chosen until either h gets too small or **Proj** successfully projects the corresponding tangent point onto M. If h gets to be smaller than some minimum acceptable stepsize, the algorithm stops with an error return. Otherwise, once the first intermediate point $x_{temp} \in M$ is found, further continuation steps are taken with the successful stepsize and in the same direction **t** until a point x is reached where the sum of the steps exceeds the original value of h. This x becomes the desired new point to be added to the database together with the simplex it completes. Then h is reset to its original value.

The following algorithm summarizes the entire triangulation process.

```
PITMAN: Input: Start point $x_0$, suggested step $h$, minimum gap angle $amin$,
            total number of bounding hyperplanes $n_h$;
    $DF(x_0)^T = (Q_1, Q_2)\binom{R}{0}$   /*Find the QR decomp. of $DF(x_0)$.*/
    $x_m$ :=Proj[$x_0, x_0, x_0, R, Q_1, Q_2, 1$]   /*Project $x_0$ onto $M$.*/
```

```
dmin :=Chkbnd[$x_m, n_k$]   /*Check if $x_m \in M_0$, i.e. if $dmin > 0$.*/
if '$dmin < 0$'  then 'error return'  endif
nod(1,7) := −1   /*Label $x_m$ as a frontal point.*/
Compute the neighborhood of six simplices of $x_m$.
Remove $x_m$ from the front.
while 'Front is non-empty'
    Get the next frontal point $x_c$.
    $DF(x_c)^T = (Q_1, Q_2)\binom{R}{0}$   /*Find the QR decomp. of $DF(x_c)$.*/
    Call Agap[$x_c, Q_2$]   /*Find gap and open edges $x_1, x_2$ at $x_c$.*/
    if '$gap < amin$' then
        Call Merge[$x_c, x_1, x_2$]   /*Identify the open edges.*/
    else
        if '$x_c$ already has 6 incident simplices' then
            'error return'.
        else
            Find the optimum number of sectors $k$ in $gap$.
            if '$k = 1$' then
                Call Addsim[$x_c, x_1, x_2$]   /*Insert one simplex.*/
            else
                Insert $k$ simplices into the gap.
            endif
        endif
    endif
    Remove $x_c$ from the list of frontal points.
end while
Output: Points and simplices which form a triangulation of $M_0$.
```

5. Numerical Experiments

The algorithm PITMAN described in the last section has been implemented in Fortran 77. We present here some sample problems run with this code.

As first example we consider the intersection of the unit sphere in R^3, defined implicitly by

$$F(x) = x_1^2 + x_2^2 + x_3^2 - 1 = 0,$$

with the half space

$$S = \{\ x \in R^3 \mid x_3 \geq -0.8\ \}.$$

Figure 5.1 shows a view of the triangulation computed by PITMAN using the stepsize $h = 0.3$, with a rotation that indicates the truncating effect of the hyperplane. The algorithm's way of handling local overlap causes four "seams" of elongated triangles on the sphere. This could be remedied by a Delaunay improvement, which is a topic of ongoing work.

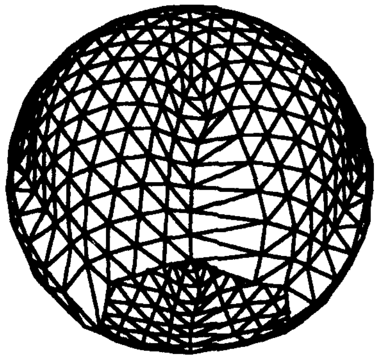

FIGURE 5.1

The second example arises as a finite-element model of the deformation of a thin, shallow circular arch. This test problem has been used by several authors and can be traced back to [W69]. We use the same model formulation as in [R86]. In an (r,θ)-polar coordinate system with the r-axis as vertical direction, the unloaded configuration of the arch is represented by the segment $\{\ (r,\theta)\ |\ r=10,\ \ -\theta_0 \le \theta \le \theta_0 = 15^\circ\}$. Let u and w be the radial and axial displacements. For pinned ends, the dimensionless total potential energy and associated boundary conditions are given by

$$\int_{-\theta_0}^{\theta_0}\left[\left[(w'-u)+\alpha_0(u')^2\right]^2+\alpha_1(u'')^2-\alpha_2\,p\,u\right]d\theta,$$
$$u(\theta)=w(\theta)=u''(\theta)=0,\quad \theta=\pm\theta_0,$$

where the primes denote derivatives with respect to θ. Here $p=p(\theta)$ is the dimensionless radial load and α_0, α_1, α_2 are dimensionless constants.

As in [R86] we use the finite element approximation consisting of a uniform mesh with eight elements. The problem was run with the following load function

$$p(\theta)=\begin{cases}\mu-4\mu(\nu-\theta)/\theta_0, & \text{if } \nu-\frac{1}{4}\theta_0 \le \theta \le \nu\,;\\ \mu+4\mu(\nu-\theta)/\theta_0, & \text{if } \nu<\theta\le\nu+\frac{1}{4}\theta_0\,;\\ 0 & \text{otherwise,}\end{cases}$$

considered already in [R88], where ν and μ are control parameters. In other words, the load is a piecewise linear hat function which has the value μ at $\theta=\nu$ and is zero outside the interval of width $0.5\theta_0$ centered at ν.

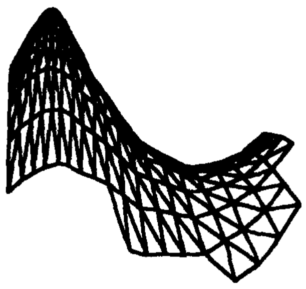

FIGURE 5.2

Figure 5.2 shows the results obtained when the initial point, computed by the continuation code PITCON (see [*RB*83]) with $\nu = 0$, is a limit point with respect to μ. Let x^c denote the dimensionless radial deformation at the center. The stepsize of the mapped triangles was $h = 0.5$, and the bounding hyperplanes were defined so as to restrict x^c to the interval $[1.1, 2.4]$ and ν to the interval $[-0.008, 0.006]$, respectively. The foldline in the (ν, μ)-plane has the shape given in Figure 5.3. Figure 5.2, which shows the manifold projected onto the (ν, μ)-plane, clearly shows a segment of this foldline that includes the local maximum and minimum points with respect to ν at $\nu = 0$ and about $\nu = 0.16$, respectively. The two-dimensional simplicial approximation algorithm [*R*88] also captured these two points, but due to the local nature of that code, two runs were needed to triangulate the manifold separately in the neighborhood of each of these two points.

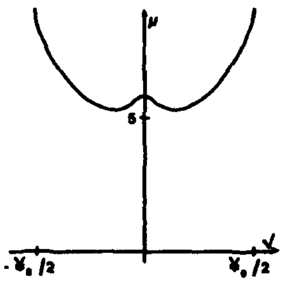

FIGURE 5.3

The third example has been used in [*MM*93] to test the robustness of their two-dimensional code. The manifold is defined as the subset

$$\left\{ (x, y, z) \in R^3 \mid F(x, y, z) = z(a^2 z^2 (b^2 - z^2) - x^2) + \epsilon = 0 \right\}.$$

When $\epsilon = 0$, the cross-section in the (x, z)-plane for any fixed value of y is the x-axis on top of a figure-eight, which is symmetric with respect to the z-axis and intersects the z-axis at $z = -b$, 0, and b. As ϵ becomes positive, the solution set in the (x, z)-plane breaks into two components: one below the x-axis and diffeomorphic to a circle, and the other the cross-section of an inverted trough above the x-axis and diffeomorphic to the real line. The size of ϵ determines the width of the trough opening, with larger values of ϵ giving larger widths. When the variable y is added to the problem the result is a two-component manifold, invariant in y, whose second component is an inverted trough which is above the plane $z = 0$ and whose opening is parallel to the y-axis. A narrow trough opening is challenging because the solver may jump across it without exploring the trough itself.

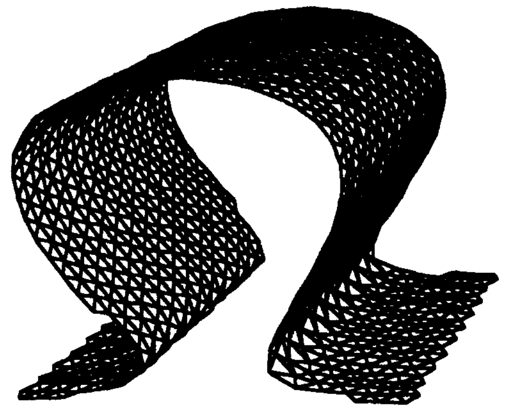

FIGURE 5.4

Figure 5.4 shows the results of the triangulation algorithm when $a = 4.0$, $b = 0.25$, $\epsilon = 5.0 \times 10^{-5}$, the stepsize is $h = 0.03$, and the initial point $(x, y, z) = (0, 0, 0.25)$. The bounding hyperplanes were defined to restrict x to the interval $[-0.15, 0.15]$ and y to the interval $[-0.14, 0.14]$. PITMAN followed the curvature without falsely jumping across the opening of the trough.

References

[AS84] Allgower, E. L. and Schmidt, P. H.: Piecewise-linear Approximation of Solution Manifolds for Nonlinear Systems of Equations, in Lecture Notes in Econ. and Math. Systems, Vol 226, Springer Verlag, Heidelberg 1984, pp 339-347

[AS85] Allgower, E. L. and Schmidt, P. H.: An Algorithm for Piecewise-linear Approximation of an Implicitly Defined Manifold, SIAM J. Num. Anal. 22, (1985), 322-346

[AS87] Allgower, E. L. and Gnutzmann, S.: An Algorithm for Piecewise-linear Approximation of Implicitly Defined Two-dimensional Surfaces, SIAM J. Num. Anal. 24, (1987), 452-469

[BE92] Bern, M. and Eppstein, D.: Mesh Generation and Optimal Triangulation, in "Computing in Euclidean Geometry" ed. by F. K. Hwang and D.-Z. Du, World Scientific Publ., 1992

[C93] Chew, L. P.: Guaranteed-Quality Mesh Generation for Curved Surfaces, in "Proc. Ninth Annual ACM Symp. on Comp. Geometry - 1993", to appear

[DH79] Deuflhard, P. and Heindl, G.: Affine Invariant Convergence Theorems for Newton's Method and Extensions to Related Methods, SIAM J. Num. Anal. 16, (1979), 1-10

[He93] Henderson, M. E.: Computing Implicitly Defined Surfaces: Two Parameter Continuation, Research Report RC 18777 (82115), IBM T. J. Watson Research Center, Yorktown Heights, NY, March, 1993

[Ho91] Hohmann, A.: An Adaptive Continuation Method for Implicitly Defined Surfaces, Research Report SC 91-20, Konrad-Zuse-Zentrum, Berlin, December, 1991

[MM94] Melville, R. and Mackey, S.: A New Algorithm for Two-dimensional Numerical Continuation, Preprint 1994, to be published

[RB83] Rheinboldt, W. C. and Burkhardt, J. V.: A Locally Parametrized Continuation Process, ACM Trans. Math. Software 9, (1983), 221-237

[R86] Rheinboldt, W. C.: Numerical Analysis of Parametrized Nonlinear Equations, J. Wiley and Sons, New York 1986

[R87] Rheinboldt, W. C.: On a Moving Frame Algorithm and the Triangulation of Equilibrium Manifolds, in "Bifurcation: Analysis, Algorithms, Applications" ed. by T. Kuepper, R. Seydel, R. Troger, ISNM Vol 79, Birkhauser, Basel 1987, pp 256-267

[R88] Rheinboldt, W. C.: On the Computation of Multi-Dimensional Solution Manifolds of Parametrized Equations, Numer. Math. 53 (1988) 165-181

[S79] Spivak, M.: A Comprehensive Introduction to Differential Geometry, Five Volumes, Second Edition, Publish or Perish, Berkeley, CA 1979

[W69] Walker, A. C.: A Nonlinear Finite Element Analysis of Shallow Circular Arches, Int. J. Solids Struct. 5 (1969) 97-107

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 4-4-94	3. REPORT TYPE AND DATES COVERED TECHNICAL REPORT

4. TITLE AND SUBTITLE ON THE COMPUTATION OF SIMPLICAL APPROXIMATIONS OF IMPLICITLY DEFINED TWO-DIMENSIONAL MANIFOLDS	5. FUNDING NUMBERS ONR-N-00014-90-J-1025 NSF-CCR-9203488
6. AUTHOR(S) Monica L. Brodzik Werner C. Rheinboldt	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Mathematics and Statistics University of Pittsburgh	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) ONR NSF	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A method is presented for the computation of a simplicial approximation covering a specified subset M_0 of a two-dimensional manifold M in R^n defined implicitly as the solution set of a nonlinear system $F(x) = 0$ of $n-2$ equations in n unknowns. The given subset $M_0 \subset M$ is the intersection of M with some polyhedral domain in R^n and is assumed to be bounded and non-empty. The method represents an extension of a local simplicial approximation process developed earlier by the second author.

14. SUBJECT TERMS nonlinear equation, manifold, simplical approximation			15. NUMBER OF PAGES 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT